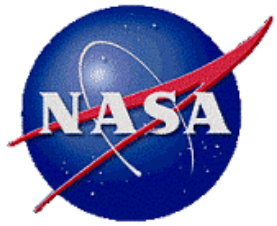


Software Engineering Research Infusion

Tom Pressburger (ARC)

Martin Feather (JPL), Lawrence Markosian (ARC),
Tim Menzies (WVU, IV&V), Luis Trevino (MSFC)

Outline

- Background
- Selected software engineering technologies

Background

- NASA Software Engineering Initiative
  - Led by the Office of the Chief Engineer
  - Improve software engineering to meet the challenges of NASA
  - Some of the areas of activity
    - Improving software development process
    - Training the workforce
    - Improving NASA guidelines, policies, procedures
    - and….

Infusing Software Engineering Research

- **Goal**
  - Transfer into practice
    - NASA-sponsored Software Engineering Research
    - Other new software engineering tools and technologies.
- **Approach**
  - Present selected technologies to the NASA software development community, and
  - Encourage and support collaborations between the researchers and NASA software developers.

Collaborations

- Initiated by a software developer interested in one or more of the technologies.
- Purpose:
  - **benefit the software development project**
  - validate the research
  - Not: further develop the research
- Funding may be available for training and customer support via…

Funding for Collaborations

- Funding for several small projects available from OSMA.
  - $25K - $50K per project.
  - Customer as PI and researcher submit a collaboration proposal.
  - Proposal form will be announced in email.
  - Due: 11/21/04
  - Start: February
- We will help facilitate unfunded collaborations.

Selected Technologies

- Culled from
  - NASA-sponsored software engineering research
  - leading edge commercial tools.
- Reviewed by researchers at several centers experienced in tech transfer of software engineering research.

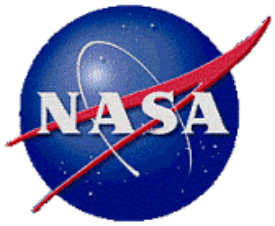
Selected Technologies (continued)

- Criteria
  - Has been successfully applied
  - Easily adopted
  - Initial focus on Software Assurance
- If initial collaborations are successful, we'll expand research product offering in FY04.

Next Step

- If you're interested in a collaboration involving a technology, contact us at http://ic.arc.nasa.gov/researchinfusion/

- We will broker matches of technology and software developers.



Selected Technologies

Lawrence Markosian

Selected Software Engineering Research Technologies

- C Global Surveyor
  - *Static analysis defect detection tool*
- Usability & Architecture
  - *Architecture design methodology*
- CodeSurfer
  - *Reverse engineering/debugging toolset*
- Perspective-based Inspections
  - *Software inspection methodology*
- Coverity SWAT
  - *Static analysis defect detection tool*
- Orthogonal Defect Classification for NASA
  - *Process improvement methodology*
- Java Path Explorer
  - *Testing tool*

Technology Description Format

- What is it
- What problem does it solve
- Features
- Successes
- Collaboration

C Global Surveyor

G. Brat & A. Venet, Automated Software Eng Group, NASA ARC
Funding: Code R, Communications, Information, and Computing Technology Program, Intelligent Systems Project

- **What is it**
  - Static analysis tool for finding defects in C applications
  - Based on “abstract interpretation”
- **What problem does it solve**
  - Fast, precise code analysis to detect defects that are hard to find through testing
    - Out of bounds array access

    - Undefined variables

A badly initialized variable caused Mars Polar Lander to crash on Mars

C Global Surveyor (continued)

- **Features**
  - Scalable to at least 600K LOC applications
  - Precise alias analysis
  - Complete path coverage
  - Very low false positive rates
  - Can detect specific classes of errors in flight software with expected 90% reduction in testing required for these errors
  - Analysis can be tailored to specific software architecture
- **Successes: Identified many undefined variables in**
  - 130K SLOC Mars Pathfinder in 45 min (for the complete system)
  - 280K SLOC DS-1 code in 2 hours (also for the complete system)

C Global Surveyor (continued)

- **Collaboration**
  - Need compilable C source code; complete build desirable but not necessary
  - Tool development team:
    - Some customization to target application architecture
    - 1 – 2 customer site visits for customization, installation, training
    - Customer email & telephone support
  - Application development team:
    - Integrate tool into implementation & testing phases of development
    - Analyze results & compare to existing practice
    - Feedback to tool development team

Usability & Architecture

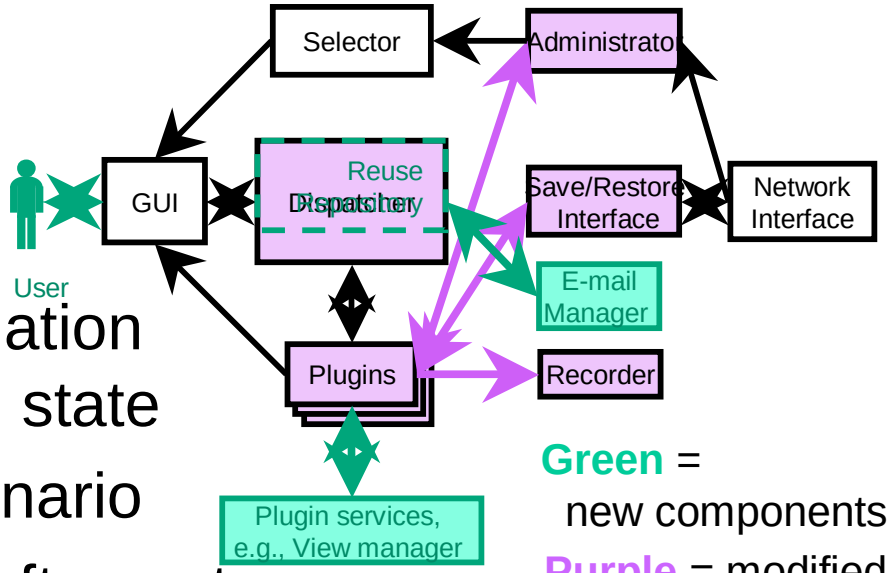
Bonnie John, CMU; Len Bass, SEI
Funding: Code R, Engineering for Complex Systems and Communications, Information, and Computing Technology programs, High Dependability Computing

- **What problem does it solve?**
  - Reduce risk that the *software architecture* of an *interactive system* has to be changed due to usability concerns.
  - "Yikes! You mean we CAN'T CANCELL COMMANDS??!!"

Oh no!

Usability & Architecture (continued)

- **What is it?**
  Methodology with—
  - 27 usability scenarios:
    - e.g., cancellation, information reuse, observing system state
    - Benefits of including scenario
    - Responsibilities of the software to support the scenarios
  - Methods for evaluating applicability of the scenarios
  - Architecture patterns that support the scenarios



Usability & Architecture (continued)

- **How to use it**
  - At architecture design (or redesign) time:
    - Consider usability scenarios
    - Decide which are important for the application
    - Ensure that the proposed software architecture fulfills responsibilities listed for those scenarios
- **Successes**
  - Modification of MERBoard's architecture, based on a usability analysis.

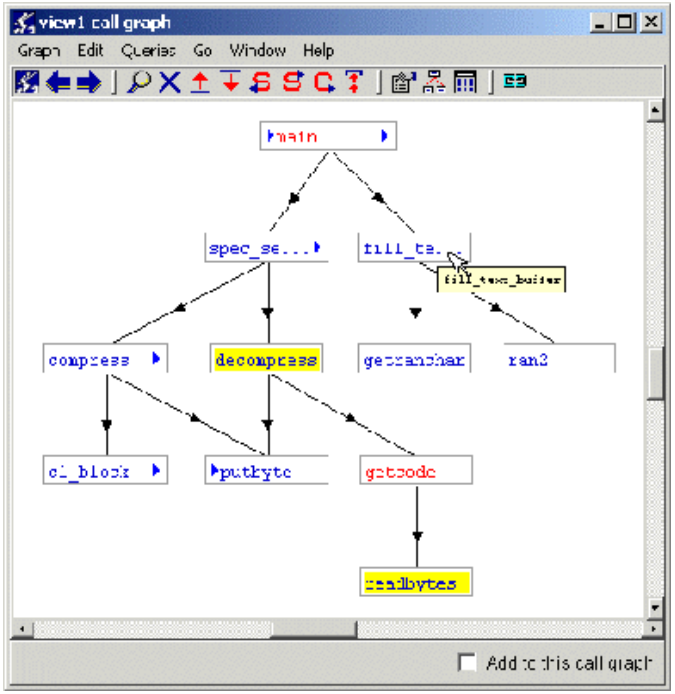
Usability & Architecture (continued)

- **Collaboration**
  - Methodology development team:
    - For one project, 2 consultants to visit over 3-day period
    - For additional collaborations, support needed to develop a handbook.
  - Application development team:
    - Evaluate user scenarios and select relevant ones
    - For relevant scenarios, work with authors to evaluate application's architecture to ensure that it supports those scenarios.
    - Work with authors to develop additional user scenarios, if any arise during the collaboration

CodeSurfer

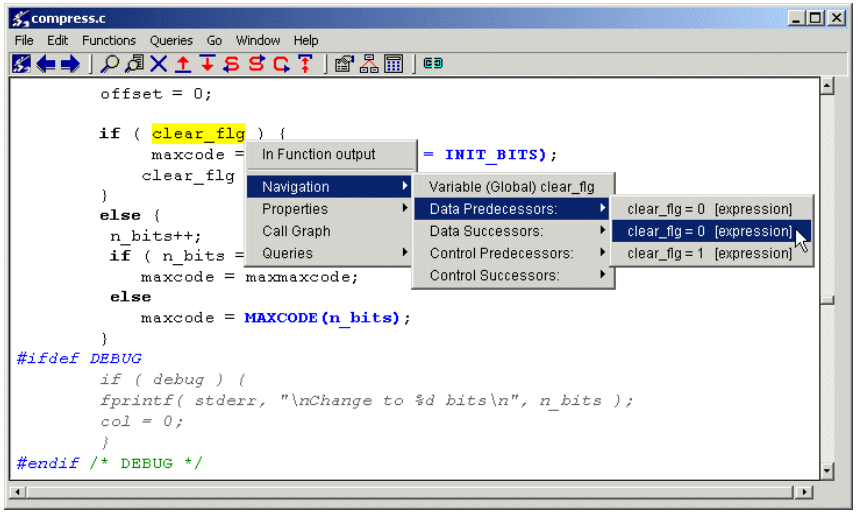
Developed by Grammatech, Inc.

- **What is it: C Source code analysis tool using—**
  - Program Slicing
    - Highlights code relevant to understanding a particular issue
    - Does impact analysis
  - Pointer Analysis
    - Tracks loads and stores via pointers
    - Takes indirect function calls into account
  - Buffer overrun detection (with plug-in)
- **What problem does it solve: More efficient—**
  - Reverse engineering
  - Debugging
  - Safety/Security auditing
  - Documentation



CodeSurfer (continued)

- **Successes**
  - Mitre, MIT, Thales, Network Associates
  - Recently obtained by NASA MSFC, not yet evaluated



  - One user (at a large aerospace company) reports:

    “Without CodeSurfer, [manual analysis of defect root cause] required about 2 to 5 days full time for one person. When using CodeSurfer, the same task has been reduced to 2 hours.”

CodeSurfer (continued)

- **Features**
  - Generates interactive, graphical reports
    - Trace data flow backward and forward through code
    - Display what variables a pointer can point to
    - Highlight code that affects selected statement(s) and/or variable(s)
    - Call graph
    - Change impact analysis, etc.
  - API for customization and batch processing
  - Commercially supported product (CodeSurfer base product, not including buffer overrun detection plug-in)
  - Approx. $2000 for single-seat floating license with 1st yr maintenance contract (without API)

CodeSurfer (continued)

- **Collaboration**
  - Need compilable C source code; build application with CodeSurfer.
  - C++ version planned for release this calendar year.
  - Best applied on applications of up to 100K – 500K LOC.
  - Tool vendor:
    - Provide training and consulting as necessary (for customization)
  - Application development team:
    - Integrate tool into implementation & testing phases of development
    - Analyze results & compare to existing practice
    - Feedback to vendor

Perspective-based Inspections

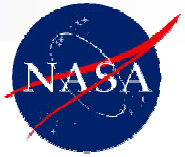
Forrest Shull et al., Fraunhofer Center
Funding: Office of Safety and Mission Assurance, Software Assurance Research Program

- **What is it**
  - Methodology for optimization inspections by focusing reviewers' roles.
- **What problem does it solve**
  - Finding defects cheaper, earlier
  - Standard inspection methods: finds around 65% of defects
  - Perspective-based reading: add up to 35%
  - 65% * 1.35 = 85.8%

Perspective-based Inspections (continued)

- **Successes**
  - JPL's Keck Interferometer
    - Improved quality of a reusable class library
    - Got inspections re-introduced on the project
    - "… we've done formal software reviews in the past, and this variation addresses one of the fundamental difficulties we experienced, by allowing folks to focus on specific topics during the review process."
      -- Tim Lockhart, Keck Software Team Lead
  - Other work on Swift BAT and James Webb Space Telescope (JWST)

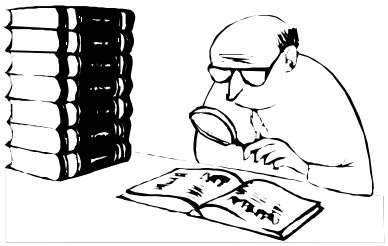
Perspective-based Inspections (continued)



- **Features**
  - Can be applied to requirements, code and other software artifacts
  - No expensive tools required
  - Perspective-based reviewers “stand in” for specific stakeholders, such as designers or testers.
- **Collaboration**
  - Methodology development team
    - Provide tutorial materials, train the trainer sessions
    - Help tune reading methods to the artifacts being read and the business goals of the system
  - Application development team:
    - Participate in training sessions
    - Consult with developers
    - Conduct inspections! Apply the methodology!

Coverity SWAT

Developed by Coverity, Inc.



- **What is it**
  - C Source code defect detection tool combining static analysis with statistical analysis to yield low false positive rate (typically 1 FP for every 5 real bugs detected).
  - Versions for C++ and Java to be released calendar year 04
- **What problem does it solve**
  - Fast, precise source code analysis to detect a wide range of defects that are hard to find through testing.
- **Successes**
  - Found over 2000 defects in Linux kernel, including many security holes, which were fixed by the developers.
  - Customers include Handspring and VMware
  - Small-scale test on an ISS application from MSFC
    - Found previously-unrecognized bugs

Coverity SWAT (continued)

- **Features**
  - Turn-key detection of common defects
    - Memory corruption
    - Resource leaks
    - Array/buffer overrun
    - NULL pointer dereferences
    - Error handling bugs
    - Security holes ...
  - Scales to millions of LOC
  - Extensible architecture and API for creating custom analyses

Coverity SWAT (continued)

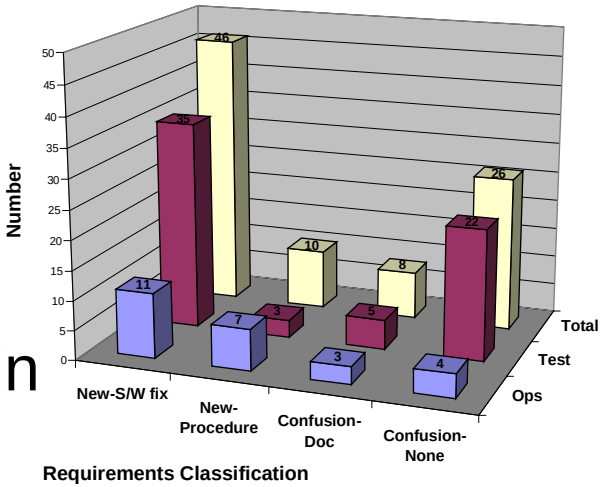
- **Collaboration**
  - Need compilable C source code; complete build desirable but not necessary
  - Vendor:
    - Provide product with maintenance & user documentation
    - Provide training and consulting as appropriate for extending the tool
  - Application development team:
    - Integrate tool into implementation & testing phases of development
    - Analyze results & compare with existing practice
    - Feedback to vendor

Orthogonal Defect Classification

Applied to NASA software development by Robyn Lutz & Carmen Mikulski, JPL

Funding: Office of Safety and Mission Assurance, Software Assurance Research Program; and National Science Foundation

- **What is it**
  - Method for analyzing software bugs to determine patterns and improve software development process
  - First developed ~1990 by Ram Chillarege at IBM, now widely used in industry
  - When faults are first seen: record “activity” and “triggering event”
  - When faults are fixed, record “target” and “type” of fix



Orthogonal Defect Classification (continued)

- **What problem does it solve**
  - Learning lessons from defect logs
    - Currently: defect logs in many incompatible formats
    - With ODC: generalized schema for defect logs
  - Provides quantitative basis for process improvement
    - Establishes a baseline for patterns of software defects
    - Much less expensive than root-cause analysis
  - Provides guidance in allocating funds for post-launch maintenance
  - Enables effective corporate memory

Orthogonal Defect Classification (continued)

- **Successes**
  - Surprises in defect logs of 8 NASA deep-space missions
    - Major anomalies in flight software were due to ground software
    - Pattern: Hardware broke; software had to compensate for loss (what broke wasn't what got fixed).
  - Recommendations to MER and future projects
    - If software's behavior confuses testers, enhance documentation (avoid similar confusion for operators)
    - Earlier testing of fault protection
  - Next generation JPL-wide problem reporting system will record data required for ODC
  - Adopted by companies such as IBM, Motorola, Telcordia, Cisco, and Nortel

Orthogonal Defect Classification (continued)

- **Features**
  - Language, platform independent
  - Produces customizable Excel graphs
  - Much local expertise
  - Useful to single project or to organization
- **Collaboration**
  - Work with JPL's ODC consultants
  - Process developer will provide training materials, presentations, provide email/telephone support
  - Software developers:
    - Change defect logger OR
    - Manual translation of existing logs to ODC format (methodology at JPL for doing this) OR
    - Auto-Map existing defect logs to similar ODC categories

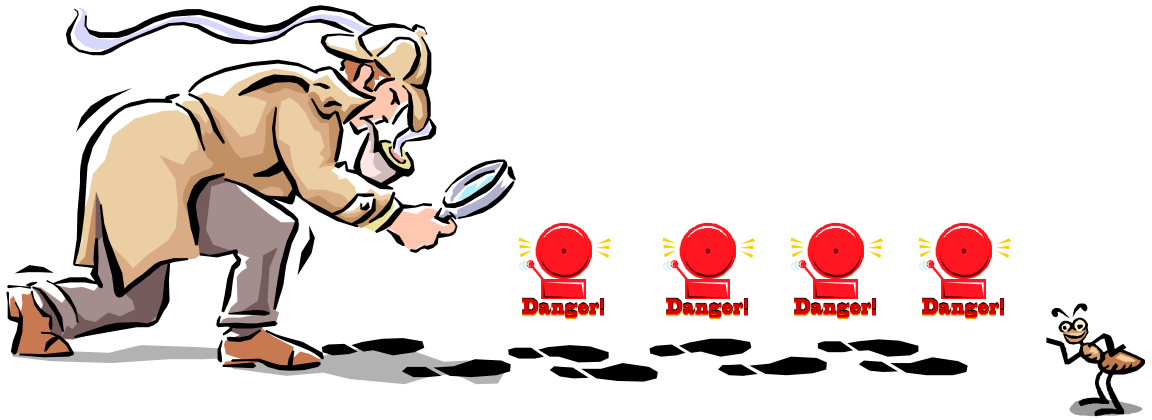
Java Path Explorer (JPaX)

K. Havelund and A. Goldberg, NASA Ames Research Center, Automated Software Engineering Group
Funding: Code R, Communications, Information, and Computing Technology Program, Information Technology Strategic Research Project

- **What is it?**
  - Code instrumenter for Java to output event traces
  - Trace analyzer
- **What problem does it solve?**
  - Finds deadlocks and data races in concurrent Java programs
  - Extension: finds violations of temporal requirements, including real-time properties, for systems that react to events in their environment
    - e.g., if a spacecraft is commanded to start an engine, it does so within 10 ms

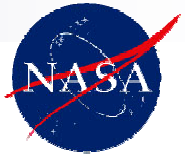
JPaX (continued)

- **Features**
  - Deadlock detection scales to large programs
  - From *one test case*, JPaX can infer concurrency problems in other executions, *even if the problem does not occur in this test*



JPaX (continued)

- **Successes**
  - Peter Gluck, JPL:
    - JPaX was run on DS-1 (Deep Space 1's attitude control system) rewritten in Java
    - "All of the problems found were of the type that would have taken months to find because they were intermittent."
    - "Found stuff we didn't know about that improved the robustness of the application."
  - Controlled study compared with conventional testing & model checking at NASA Ames:
    - Found all but one known concurrency bugs in K9 Rover real-time executive.



JPaX (continued)

- **Collaborations**
  - For concurrency analysis (this is the primary focus of the collaboration):
    - Tool developers will provide user documentation, training, consulting support.
    - Software development team applies tools, analyzes results.
      - Automatically instrument the code to emit events
      - Run a test case
      - Run analysis of event log
  - For temporal & real-time properties, software development team to work with tool developers to identify and formalize properties.

Next Step

- If you're interested in a collaboration involving a technology, contact us at http://ic.arc.nasa.gov/researchinfusion/

- We will broker matches of technology and software developers.

Selected Software Engineering Research Technologies

- C Global Surveyor
  - *Static analysis defect detection tool*
- Usability & Architecture
  - *Architecture design methodology*
- CodeSurfer
  - *Reverse engineering/debugging toolset*
- Perspective-based Inspections
  - *Software inspection methodology*
- Coverity SWAT
  - *Static analysis defect detection tool*
- Orthogonal Defect Classification for NASA
  - *Process improvement methodology*
- Java Path Explorer
  - *Testing tool*